Complejidad Temporal, Estructuras de Datos y Algoritmos

Enunciado Trabajo Final

Consideraciones Generales

El objetivo de este trabajo es integrar los contenidos vistos en la materia y el mismo se deberá realizar en forma individual.

El trabajo integrador deberá defenderse en dos exposiciones. En la primera, se presentará una versión limitada del alcance solicitado, según lo indicado en este documento. En la segunda exposición, se mostrará la versión que cumple con el alcance completo. Las fechas límite para la defensa de la primera parte y del trabajo en su totalidad están publicadas en el aula virtual, en el enlace “Plan de trabajo”.

El trabajo se presentará junto con una presentación que debe incluir:

- Datos del autor del trabajo final.
- Un diagrama de clases UML describiendo la estructura del sistema, mostrando las clases del sistema, sus atributos, métodos y las relaciones entre los objetos.
- Detalles de implementación, problemas encontrados y como fueron solucionados, condiciones de ejecución, etc.
- Las imágenes de todas las pantallas que componen el sistema codificado junto con una descripción completa de las mismas.
- Ideas o sugerencias para mejorarlo o realizar una versión avanzada del mismo.
- Una breve conclusión o reflexión de la experiencia adquirida a partir de la realización del trabajo final.

El trabajo deberá defenderse en una exposición presencial.

La presentación será evaluada tanto por su contenido como por la forma en que se exponga, y su calificación impactará en la nota final del trabajo

Enunciado

Machine Learning o **Aprendizaje Automatizado** tiene por objetivo desarrollar técnicas que permitan que las computadoras aprendan. Más concretamente, se trata de crear programas capaces de generalizar comportamientos a partir de la información suministrada.

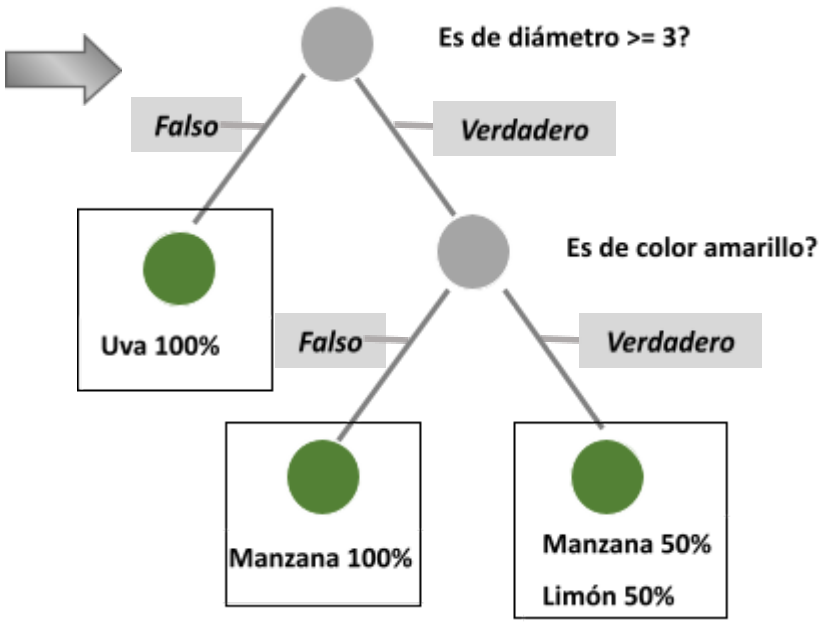
De entre todas las posibles herramientas utilizadas en el área de *machine learning*, una de las que más se destaca son los árboles de decisión, que proporcionan un conjunto de reglas que se van aplicando sobre conjuntos de datos para decidir qué predicción es la más adecuada según sus atributos.

Un árbol de decisión está formado por un conjunto de nodos de decisión (interiores) y de nodos-hojas:

- Un nodo de decisión está asociado a un atributo y de él parten dos ramas, cada una de ellas representa los posibles valores (Verdadero o Falso) que puede tomar el atributo asociado. De alguna forma, un nodo de decisión es como una pregunta que se le hace al conjunto de datos analizado, y dependiendo de la respuesta que dé, se tomará una de las dos ramas salientes.
- Un nodo-hoja está asociado a la clasificación que se quiere proporcionar, y nos devuelve la predicción del árbol con respecto al conjunto de datos de entrada.

En la siguiente figura, se muestra un ejemplo en el cual se construye un árbol de decisión a partir de un conjunto de datos de entrenamiento. En la parte izquierda se encuentra una tabla que contiene en sus dos primeras columnas las características que nos interesan sobre las frutas (color y diámetro) y en la última columna las etiquetas que indican a que fruta pertenecen dichas características. El árbol de la derecha es construido a partir de la tabla antes mencionada (datos de entrenamiento) y muestra el mecanismo que se utilizará cuando se deba realizar una predicción de a que fruta (o frutas) pertenecen un conjunto de características.

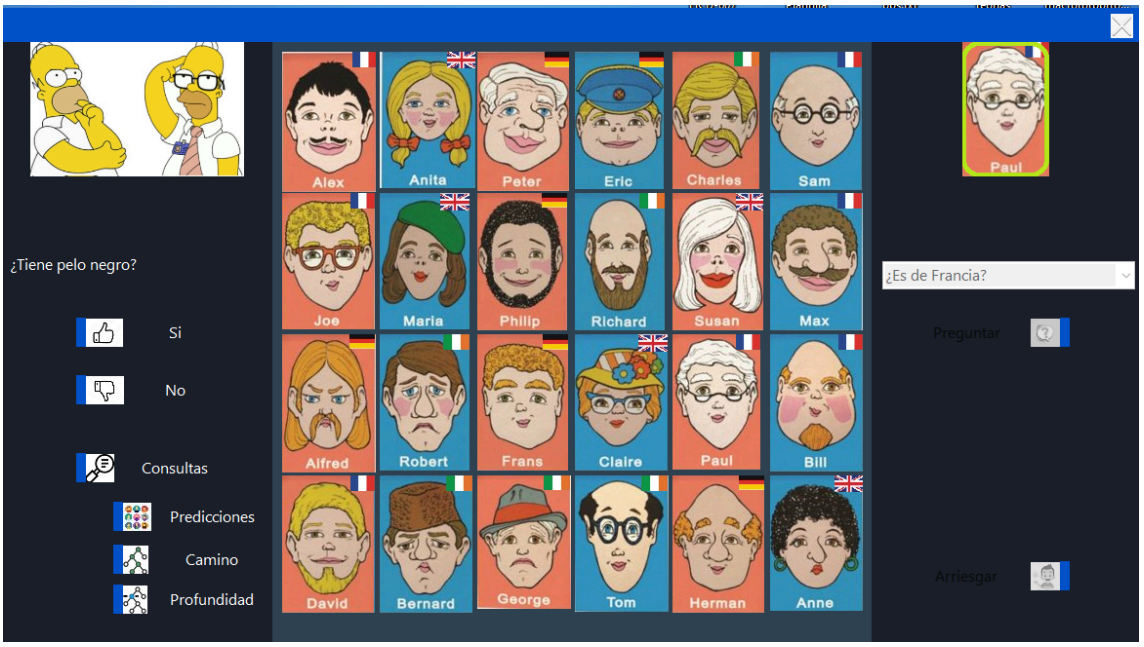
Características		Etiqueta
Color	Diámetro	
Verde	3	Manzana
Amarillo	3	Manzana
Amarillo	3	Limón
Roja	1	Uva



Nota: Observar que el árbol está formado únicamente por nodos de decisión y nodos-hojas, mientras que los rectángulos grises claros son simplemente indicaciones de la rama a tomar de acuerdo al resultado de contestar la pregunta de cada nodo de decisión.

Una empresa informática está llevando a cabo un juego de computadora llamado *Who is Who* (**WiW**) que tiene por objetivo adivinar primero cuál es el personaje elegido por el oponente. Uno de los dos jugadores es una entidad de software que posee una estrategia codificada a fin de ganar el juego (**Bot**).

WiW inicia solicitando al usuario que seleccione el directorio donde se encuentran los recursos que se utilizaran durante la partida. A continuación, se le solicitará al usuario que seleccione uno de los posibles personajes que se le muestra en pantalla para luego dar por iniciada la partida desde la siguiente vista:



El juego se desarrolla por turnos. El jugador en turno debe responder las preguntas que le realiza el otro jugador, pudiendo ser tanto el **Bot** como el usuario, a fin de poder identificar cual es el personaje elegido por el oponente. Las preguntas realizadas serán similares a:

- «¿Es rubio?» – «No»
- «Es castaño» – «Sí»
- «¿Lleva gafas?» – «No»
- «¿Es mujer?» – «No»
- «¿Tiene barba?» – «Sí» …

La empresa informática lo ha contratado a Ud. para implementar la estrategia que utiliza el **Bot** y esta debe estar basada en el uso de un **árbol de decisión**. En el juego la codificación de la estrategia antes descripta se realizará en la clase **Estrategia** a través de los siguientes métodos que Ud. debe implementar:

1. **CrearArbol** (Clasificador clasificador): Este método construye un árbol de decisión a partir de un clasificador que es enviado como parámetro y retorna una instancia de **ArbolBinario<DecisionData>**.

2. **Consulta1** (ArbolBinario< DecisionData > arbol): Retorna un texto con todas las posibles predicciones que puede calcular el árbol de decisión del sistema.

3. **Consulta2** (ArbolBinario< DecisionData > arbol): Retorna un texto que contiene todos los caminos hasta cada predicción.

4. **Consulta3** (ArbolBinario< DecisionData > arbol): Retorna un texto que contiene los datos almacenados en los nodos del árbol diferenciados por el nivel en que se encuentran.

En **la primera** de las dos exposiciones, **se deberá tener implementado y funcionando los puntos 1 y 2 del listado anterior de funcionalidades** para ser presentadas bajo los lineamientos antes descritos. En la segunda exposición, se deberá contar con el listado completo funcionando para ser presentado de manera análoga a lo realizado en la primera exposición.

La cátedra proveerá clases utilitarias a fin de simplificar la construcción y prueba del juego. A continuación, se describen los métodos más importantes:

Clase	**Métodos**
DecisionData	– **DecisionData**(Pregunta question): Construye un objeto que almacena una pregunta. – **DecisionData**(Dictionary<string, int> predictions): Construye un objeto que almacena un dato de un nodo-hoja de tipo **Dictionary<string, int>**. – **ToString():** Devuelve un texto que contiene el dato almacenado en formato **string** .
Clasificador	– **crearHoja**(): Retorna ***True*** si el conjunto de datos corresponde a un nodo-hoja y ***False*** en caso contrario. – **obtenerDatoHoja** (): Devuelve el dato que debe almacenarse en un nodo-hoja. El dato devuelto es de tipo **Dictionary<string, int>** donde la clave del diccionario es la etiqueta y el valor es la cantidad de ocurrencias. – **obtenerClasificadorIzquierdo** (): Devuelve el clasificador correspondiente a la rama izquierda. – **obtenerClasificadorDerecho** (): Devuelve el clasificador correspondiente a la rama derecha. – **obtenerPregunta** (): Devuelve la pregunta que debe almacenarse en un nodo de decisión a partir del conjunto de datos con el cual se creó el clasificador.